**TOPICS**

**Maximum Likelihood Estimation**

**Introduction**

One of the main techniques in statistics and data analysis is Maximum Likelihood Estimation (MLE). Through the use of this technique, we may estimate a statistical model's parameters and make sure that our model most closely matches the observed data. However, how does it operate and why is it so popular? Let's dissect it.

**Mathematical and Technical Intuition**

MLE's fundamental goal is to identify the values of specific factors that increase the likelihood that the observed data is real. Consider attempting to estimate a cat's weight only based on appearance. Even if you can't guess it precisely, you can get a good idea. MLE performs a comparable function in a more advanced manner. Finding the parameter values that would make a collection of observations more likely requires taking a set of data and a model that explains how those observations were obtained.

**Presentation of Ideas**

A few essential phases make up the MLE process:

1. **Select a Model:** Any statistical model that you think adequately describes your data might be used for this.
2. **Create the Likelihood Function:** This function tells us, for a given set of parameter values, the likelihood that we will see our data.
3. **Determine the Maximum:** We make adjustments to the parameters until the likelihood function peaks.

**Sample Code**

Here’s a Python example using the scipy library:

import numpy as np

from scipy.optimize import minimize

# Assume we have some data

data = np.array([...])

# Define the likelihood function

def likelihood(params):

# Parameters of the model

mu, sigma = params

# Probability of observing the data

prob\_data = np.prod((1 / (sigma \* np.sqrt(2 \* np.pi))) \* np.exp(-((data - mu)\*\*2) / (2 \* sigma\*\*2)))

# We take the negative log-likelihood

return -np.log(prob\_data)

# Perform MLE

result = minimize(likelihood, [0, 1], bounds=[(-10, 10), (0.01, 10)])

mu\_mle, sigma\_mle = result.x

**Related Examples**

MLE isn’t just a theoretical concept; it has practical applications across various fields:

* In **finance**, MLE helps in modeling stock returns to predict future prices.
* **Epidemiologists** use MLE to estimate the spread rate of diseases.
* In **machine learning**, MLE is used to train models like logistic regression.

**Conclusion**

One of the most useful tools in a statistician's toolbox is the maximum likelihood estimate. It serves as a link between unprocessed data and insightful understandings, enabling us to measure uncertainty and arrive at wise conclusions. Knowing MLE is a first step towards becoming an expert in data analysis, regardless of experience level or interest.

**Maximum A Posterior**

**Maximum a Posteriori (MAP) Estimation: A Bayesian Approach**

The use of Maximum a Posteriori (MAP) estimate is essential in the field of statistical modelling. It's an effective method that estimates model parameters by fusing observed data with past knowledge. Let's examine MAP's specifics and see how it varies from other estimating techniques.

**Density Estimation**

Let's use density estimate to establish the scene before we go into MAP. Let's say we have an observation dataset, represented by the letter (X). Estimating the joint probability distribution for this dataset is our aim. It can be difficult to estimate the full distribution explicitly, though. Rather, we frequently accept a point estimate, like the mean.

There are two popular methods for estimating density:

1. A frequentist technique called maximum likelihood estimation (MLE) seeks to identify the parameter values that maximise the chance of witnessing the sample data.
2. Maximum a Posteriori (MAP): A Bayesian technique that considers our preconceived notions about the model in addition to the likelihood of the data..

**What Is MAP?**

MAP estimation provides an alternative to MLE by incorporating prior information. Here’s how it works:

1. **Bayesian Framework:** MAP is a probabilistic framework that takes into account a prior probability or belief about the model in addition to the likelihood of witnessing the data given a model. It strikes a balance between our preconceived notions and the observable facts (probability).
2. **Posterior Distribution**: The main concept is to compute the conditional probability, weighted by our previous opinion about the model, of witnessing the data given a model. In terms of math, it is stated as:

[ \text{MAP}(\theta) = \arg\max\_\theta P(\theta | \text{data}) = \arg\max\_\theta \frac{P(\text{data} | \theta) \cdot P(\theta)}{P(\text{data})} ]

* + The model parameters are denoted by (\theta).
  + The posterior distribution of (\theta) in light of the data is (P(\theta | \text{data})).
  + The probability of the data provided (\theta) is (P(\text{data} | \theta)).
  + The prior distribution that reflects our previous assumptions about (\theta) is (P(\theta)).

1. **Optimization**: The posterior distribution is maximised when we determine the value of (\theta). This calls for the solution of an optimisation issue.

**Solving for (\theta)**

1. **Observing Data**: We start with a sample of independent and identically distributed (i.i.d.) random variables.
2. **Likelihood Same as MLE**: We assume the likelihood function is the same as in MLE.
3. **Prior Distribution**: We introduce a prior distribution for (\theta), denoted as (P(\theta)).
4. **MAP Estimator**: The MAP estimator of (\theta) is the value that maximizes the posterior distribution:

[ \theta\_{\text{MAP}} = \arg\max\_\theta \left(\log P(\text{data} | \theta) + \log P(\theta)\right) ]

**Interpretation**

* MAP weighs the data's probability against our preconceived notions.
* In situations where we have less data, it offers a more reliable estimate.
* The MAP estimate is represented by the mode of the (\theta) posterior distribution.

In a nutshell MAP estimation is a useful tool in many machine learning algorithms since it enables us to include previous information into our model.

Recall that MAP adopts a more comprehensive strategy by taking into account both the evidence and prior beliefs, whereas MLE just considers the likelihood. It's an addition to our statistical toolkit, a Bayesian twist.

**Related Examples**

1. Signal processing: MAP applies past knowledge of signal characteristics to extract the most likely signal from noisy data.
2. Medical Imaging: By utilising previous knowledge about typical anatomical features, MAP enhances picture clarity in medical imaging.
3. Finance: MAP uses a combination of recent market data and historical patterns to update the predicted return on assets.
4. Sports Analytics: By weighing a team's historical results against the most recent match data, MAP calculates the likelihood of a team winning.

These brief examples show how MAP estimate uses past information to help decision makers in a variety of fields make better choices.

**Bayesian Inference**

**Introduction**

Among the frameworks used in statistics and data analysis, Bayesian inference is particularly potent and adaptable. It enables us to revise our opinions about a situation or theory in light of fresh information. Making educated decisions requires a grasp of Bayesian inference, regardless of whether one is a data scientist, researcher, or decision-maker.

[**What Is Bayesian Inference?**](https://www.askpython.com/python/examples/bayesian-inference-in-python)

When new information becomes available, we may use the approach of Bayesian inference to revise our views about an event or hypothesis. It's similar to modifying your forecasts in light of new information. The main premise is this:

* Prior Probability: We begin with a preliminary belief (prior) on the parameter or event.
* Likelihood: Using our past knowledge, we determine the likelihood of newly observed data.
* Posterior Probability: To obtain a more precise posterior belief, we update our prior using the Bayes theorem.

[**Bayes’ Theorem**](https://www.askpython.com/python/examples/bayesian-inference-in-python)

Bayes’ theorem is the foundation of Bayesian inference. It relates the prior, likelihood, and posterior probabilities:

[ P(\text{Hypothesis} | \text{Evidence}) = \frac{P(\text{Evidence} | \text{Hypothesis}) \cdot P(\text{Hypothesis})}{P(\text{Evidence})} ]

* (P(\text{Hypothesis} | \text{Evidence})): Posterior probability (updated belief).
* (P(\text{Evidence} | \text{Hypothesis})): Likelihood (probability of observing evidence given the hypothesis).
* (P(\text{Hypothesis})): Prior probability (initial belief).
* (P(\text{Evidence})): Marginal likelihood (normalizing constant).

[**Implementing Bayesian Inference in Python**](https://www.askpython.com/python/examples/bayesian-inference-in-python)

Let’s implement Bayesian inference using Python. We’ll estimate the mean of a normal distribution. Here’s a step-by-step guide:

Step 1: Generate Synthetic Data

import numpy as np

import matplotlib.pyplot as plt

# Generate synthetic data

np.random.seed(42)

true\_mu = 5

true\_sigma = 2

data = np.random.normal(true\_mu, true\_sigma, size=100)

Step 2: Define Prior Hyperparameters

# Define prior hyperparameters

prior\_mu\_mean = 0

prior\_mu\_precision = 1 # Variance = 1 / precision

prior\_sigma\_alpha = 2

prior\_sigma\_beta = 2 # Beta = alpha / beta

**Step 3:** Update **Prior with Data**

# Update prior hyperparameters with data

posterior\_mu\_precision = prior\_mu\_precision + len(data) / true\_sigma\*\*2

posterior\_mu\_mean = (prior\_mu\_precision \* prior\_mu\_mean + np.sum(data)) / posterior\_mu\_precision

posterior\_sigma\_alpha = prior\_sigma\_alpha + len(data) / 2

posterior\_sigma\_beta = prior\_sigma\_beta + np.sum((data - np.mean(data))\*\*2) / 2

Step 4: Calculate Posterior Parameters

# Calculate posterior parameters

posterior\_mu\_variance = 1 / posterior\_mu\_precision

posterior\_sigma\_variance = posterior\_sigma\_alpha / posterior\_sigma\_beta

**Real-World Example: Predicting Website Conversion Rates**

Let's forecast website conversion rates using Bayesian reasoning. Watch this space for the full example!

* Medical Diagnosis: Revising the likelihood that a patient is ill in light of fresh test findings.
* Part-of-speech tagging and language modelling in natural language processing: using Bayesian models.
* A/B testing: Using Bayesian approaches to estimate conversion rates for various website versions.
* Hyperparameter tuning for machine learning: Bayesian optimisation to identify ideal hyperparameters.

**Bayesian Inference vs. Maximum Likelihood vs. Frequentist**

Whereas frequentist approaches just consider data, Bayesian inference enables us to take previous information into account. A frequentist method that maximises the likelihood function is called maximum likelihood estimation (MLE).

**GENERATIVE MODELS – VARIATIONAL AUTOENCODERS (VAEs)**

**What Are VAEs?**

A family of generative models known as variational autoencoders (VAEs) combines concepts from deep neural networks, information theory, and statistics. They are made to effectively address the high-dimensional data production challenge. Virtual Autoencoders (VAEs) translate input data into the parameters of a probability distribution, as opposed to standard autoencoders that map input data onto a latent vector. They specifically model in the latent space the mean and variance of a Gaussian distribution.

**Key Concepts**

**A decoder and an encoder**

**Encoder:** An input observation, such as an image, is mapped to the latent distribution's parameters via the encoder network**.** In the latent space, these parameters stand for the Gaussian distribution's mean and variance.

**Decoder: Data is produced from the latent space by the decoder network.** It restores the latent variables to the original data domain (for example, using the latent representation to rebuild an image).

**Regularization**:

The regularisation of VAEs is the main novelty. To guarantee desired qualities, the encodings distribution (latent space) is regularised during training. VAEs promote the latent space to have meaningful representations that make data production easier by setting limits on it. This regularisation and variational inference in statistics are closely related, which is where the word "variational" comes from.he key innovation in VAEs lies in their regularization. During training, the encodings distribution (latent space) is regularized to ensure desirable properties.

By imposing constraints on the latent space, VAEs encourage it to have meaningful representations that facilitate data generation.

The term “variational” originates from the close relationship between this regularization and variational inference in statistics.

**Variational Inference**:

* A statistical method for approximating complicated probability distributions is variational inference.
* By maximising the Evidence Lower Bound (ELBO), it enables us to discover the latent space distribution in the setting of VAEs.
* During training, the ELBO serves as a stand-in objective function.

**Evidence Lower Bound (ELBO**):

* Two elements are balanced by the ELBO.
* Reconstruction Loss: Indicates how successfully the input data is rebuilt by the model.
* Term for Regularisation: Promotes a submissive latent space.
* VAEs learn meaningful representations and facilitate data creation by optimising the ELBO.

**Practical Implementation**

Consider using a face dataset to train a VAE. The generative capabilities of the model are demonstrated by sampling from the latent space after training. To put a VAE into action, do the following:

* Describe the networks for encoders and decoders.
* Establish the reconstruction loss and regularisation term in the loss function.
* Utilise datasets (like MNIST) to train the model and observe how effective VAEs are in producing new data.

To sum up, variational autoencoders offer a probabilistic framework for producing data and learning significant latent representations. Their ability to integrate neural networks and statistical concepts makes them an effective tool in the generative modelling space.